

# Identifying Conserved Gene Clusters in the Presence of Orthologous Groups

[Extended Abstract]

Xin He  
Department of Biology  
CB#3280  
University of North Carolina  
Chapel Hill, NC 27599  
xhe2@email.unc.edu

Michael H. Goldwasser  
Dept. of Math. and Math. Computer Science  
Saint Louis University  
221 N. Grand Blvd  
St. Louis, MO 63103-2007  
goldwamh@slu.edu

## ABSTRACT

Current biological evidence suggests a correlation between the function and the position of genes in chromosomes. Examples include operon structure in prokaryotic genomes and similar expression patterns of neighboring genes in some eukaryotic genomes. In this paper, we present a new model and algorithm for identifying conserved gene clusters from pairwise genome comparison. This generalizes a recent model called “gene teams.” A gene team is a set of orthologous genes that appear in two or more species, possibly in a different order yet with the distance of adjacent genes in the team for each chromosome always no more than a certain threshold. We remove the constraint in the original model that each gene must have a unique copy in the chromosomes, and thus allow the analysis on complex prokaryotic or eukaryotic genomes with extensive paralogs. Our algorithm runs in  $O(mn)$  time and uses  $O(m+n)$  space, where  $m$  and  $n$  are the number of common genes in each chromosome. We used this approach to study two bacterial genomes, *E. coli* and *B. subtilis* and successfully identified 85 conserved clusters, including clusters containing uncharacterized genes and a large cluster consisting of 21 ribosomal proteins. Our implementation is publicly available at <http://euler.slu.edu/~goldwasser/cogteams/>.

## Categories and Subject Descriptors

J.3 [Life and Medical Sciences]: *biology and genetics*;  
F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*computations on discrete structures*; G.2.1 [Discrete Mathematics]: Combinatorics—*combinatorial algorithms, recurrences and difference equations*

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RECOMB '04, March 27–31, 2004, San Diego, California, USA.  
Copyright 2004 ACM 1-58113-755-9/04/0003 ...\$5.00.

## General Terms

Algorithms

## Keywords

bioinformatics, clusters of orthologous genes, comparative genomics, conserved gene clusters, gene teams

## 1. INTRODUCTION

A fundamental question in genomics is the relationship between a gene’s position in a chromosome and its function. The current evidence suggests that genes are not randomly distributed in the chromosomes and that genes which are physically close to each other tend to represent groups of genes with a functional relationship, even if not contiguous. In prokaryotic genomes, functionally related genes are often organized into “operons,” clusters of genes that are regulated as one transcriptional unit. It has been shown that the gene clusters whose order are conserved have a strong tendency to encode proteins that have physical interaction [2]. Recent studies also indicate that in some eukaryotic genomes, neighboring genes tend to have similar expression patterns [11]. One plausible explanation for this phenomenon is that if a set of genes remain as relative neighbors in multiple species, it is likely that there are evolutionary constraints that maintain this arrangement and thus these genes are likely to represent a functionally significant group. Such a gene group is referenced in literature as a *conserved gene cluster*. The functional coupling of genes in conserved clusters has been demonstrated computationally in studies involving multiple prokaryotic genomes [5, 9].

In a recent paper, Bergeron, Corteel and Raffinot [1] introduce a concept denoted as a *gene team*. A gene team is a set of orthologous genes that appear in two or more species, possibly in a different order yet with the distance between adjacent genes in the team for each chromosome always no more than a certain threshold. Their model includes a restriction that each gene has at most one orthologous partner in the other chromosome. They provide an  $O(n \log^2 n)$ -time,  $O(n)$ -space algorithm which identifies the gene teams common to a pair of chromosomes comprised of  $n$  common genes. The restriction to a one-to-one orthologous relation limits their gene team model, as many genomes contain a large number of multigene families, which are similar or nearly

identical genes of the same origin, resulting from gene duplication. This is especially so in higher organisms.

In this paper, we extend the notion of “gene teams” to allow any number of paralogs and orthologs. We refer to this generalization as COG teams, for reasons explained in Section 3. We provide a divide-and-conquer algorithm to find such COG teams for *pairwise* chromosome comparison. The algorithm has  $O(mn)$  running time and  $O(m+n)$  space requirement, where  $m$  and  $n$  are the number of orthologous genes in the two chromosomes.

This paper is structured as follows. In Section 2, we review some related research. In Section 3, we formalize the concept of COG teams. In Section 4, we present the algorithm of identifying COG teams of two chromosomes and the analysis of running time and space. In Section 5, we demonstrate the practical use of this algorithm by performing analysis on genomic data.

## 2. RELATED WORK

Various computational methods have been proposed to identify conserved gene clusters by comparison of multiple genomes, both experimentally and formally. Models of a conserved gene cluster vary across several axes: (1) whether the genes in a cluster are required to be immediate neighbors on a chromosome; (2) whether gene order is allowed to vary among chromosomes; (3) whether paralogs are allowed on a single chromosome, and thus a many-to-many orthologous relationship among chromosomes; (4) whether comparison is pairwise between two genomes, or generally between three or more genomes.

If disallowing paralogs and requiring immediate neighboring relationships, each chromosome can be modeled as a permutation over the set of genes. The problem of locating gene clusters is then related to a previously studied combinatorial problem, that of finding common intervals in two or more permutations. Uno and Yagiura [14] present an  $O(n+K)$ -time,  $O(n)$ -space algorithm to find all  $K$  common intervals between two permutations of  $n$  items, where  $K \leq \binom{n}{2}$ . Heber and Stoye [6] generalize this algorithm to an  $O(cn+K)$ -time,  $O(cn)$ -space algorithm to find all  $K$  common intervals between  $c \geq 2$  permutations. Recently, Didier [3] extends these approach to include paralogs by considering a more general sequence definition than a strict permutation.

One drawback of the above approaches is that a single misplaced gene disqualifies the commonality between two otherwise similar intervals. To remedy this drawback, gene clusters can be defined so long as the gaps between adjacent genes in the cluster are less than a given threshold. This is the approach taken by Overbeek *et. al.* [9], by Bergeron, Corteel and Raffinot [1], as well as by us. We note that the work of Overbeek *et. al.* considers only pairwise comparison of chromosomes, as does ours. Bergeron, Corteel and Raffinot do observe that their algorithm can be generalized to find common teams among a set of  $c$  chromosomes, resulting in an  $O(cn \log^2 n)$ -time,  $O(cn)$ -space algorithm. However, the insistence on one-to-one orthologous relations severely limits the utility of that generalization. For pairwise comparison, a common technique to ensure one-to-one orthologous relationships is to rely on the bidirectional best hit method, as in [7, 9]. The generalization of this criteria to three or more chromosomes precludes the inclusion of many likely orthologs.

## 3. MODEL AND DEFINITIONS

To define the identity of genes when multiple chromosomes are compared, we use the biological concepts of orthologs and paralogs. Orthologs are genes in different species that have the same evolutionary origin and are generated during speciation; paralogs are genes in the same species caused by the duplication of ancestor genes and the subsequent variation. In the simplest scenario, one gene in one chromosome has exactly one ortholog in another chromosome, i.e. there is a one-to-one correspondence between orthologs in two chromosomes. More commonly, a gene can have multiple orthologs in another genome, if the original ortholog of this gene created paralogs after speciation. In this case, all the paralogs in the second genome are orthologs of this gene. In general, genes of the same origin in multiple genomes form orthologous groups, which are sometimes referred to as Clusters of Orthologous Groups (COG) [12, 13]. In our work, we adopt the term COG to denote the ideal orthologous gene groups and we represent a chromosome as an ordered sequence of genes, where each gene is associated with its containing COG. It is important that we differentiate this ideal notion of a COG from that of the COG database<sup>1</sup> supported by NCBI [8], which represents only identified and sometimes unconfirmed COGs.

*Definition 1.* A *chromosome* is a pair  $C = (\Sigma, X)$  where  $\Sigma$  is a set of COGs and where  $X$  is an ordered sequence of genes. Each gene of  $X$  is denoted as a pair  $(p, c)$  with  $p \in \mathbb{R}$  representing the physical position of the gene in the chromosome and  $c \in \Sigma$  the COG to which the gene belongs. We will refer to  $\Sigma$  as the *alphabet* for  $C$ .

*Definition 2.* Given a chromosome containing two genes  $g_i = (p_i, c_i)$  and  $g_j = (p_j, c_j)$ , the distance between  $g_i$  and  $g_j$  is defined as  $\Delta(g_i, g_j) = |p_i - p_j|$ .

The sequence of genes is ordered such that the position values are monotonically increasing. However, the positions need not be consecutive nor integral. In practice, this value represents a physical measure of the gene position, such as base pair distance. Since we are interested in performing comparative genomics on a pair of chromosomes, genes which do not have orthologs in the opposite chromosome should be omitted in the representation, thereby reducing the computational costs for our algorithm.

*Example 1.* We consider a chromosome  $C$  over alphabet  $\Sigma = \{a, b, c, d, e\}$ , with genes  $(1, c), (4, e), (5, d), (6, a), (8, b), (10, e), (11, d)$ . Throughout this paper, examples with integral positions are illustrated using a textual representation, in this case  $C = \langle \mathbf{c**eda*b*ed} \rangle$ . We use asterisks to represent gaps between genes of the relevant COGs. In a pairwise chromosome comparison, such gaps may represent genes which do not have orthologs in a second chromosome.

*Definition 3.* Given chromosome  $C = (\Sigma, X)$  and any  $X' \subseteq X$ , we use the notation  $\Sigma(X') \subseteq \Sigma$  to denote the subset of COGs occurring in  $X'$ .

*Definition 4.* Given chromosome  $C = (\Sigma, X)$ , we denote as a *subsequence* any pair  $(\Sigma(X'), X')$  where  $X' \subseteq X$ .

<sup>1</sup>Of course in practice, the NCBI database offers a tangible approximation to the ideal. In fact we rely on it as such in our own experiments discussed in Section 5.

For example,  $X' = \{(4, e), (6, a), (10, e), (11, d)\}$ , forms a subsequence of chromosome  $C$  given in Example 1,

*Definition 5.* Given chromosome  $C = (\Sigma, X)$  and subsequence  $C' = (\Sigma', X')$ , we say that  $C'$  is a *subchromosome* if it is also the case that  $X'$  is a *contiguous* subset of  $X$ .

For example,  $X' = \{(5, d), (6, a), (8, b), (10, e)\}$ , when paired with  $\Sigma(X') = \{a, b, d, e\}$ , forms a subchromosome of chromosome  $C$  given in Example 1.

### 3.1 COG Teams

In this section, we introduce a series of definitions to capture the notion of neighboring genes and conserved gene clusters. Two genes  $g_i$  and  $g_j$  are considered neighboring whenever  $\Delta(g_i, g_j) \leq \delta$  for a chosen parameter  $\delta > 0$ . Most of our definitions are generalizations of those originally made by Bergeron, Corteel and Raffinot in an earlier model which assumed one-to-one orthologous relationship, [1, 7]. In generalizing these concepts, we now carefully distinguish between a collection of genes as opposed to a collection of COGs. A deeper comparison between our model and this earlier model is discussed in Section 3.2.

*Definition 6.* Given chromosome  $C = (\Sigma, X)$ , a subsequence  $C'$  is termed a  $\delta$ -subsequence of  $C$  if every pair of adjacent genes of  $C'$  are separated by positional distance of at most  $\delta$ . We further say that  $C'$  is a  $\delta$ -run if it is a maximal  $\delta$ -subsequence with respect to inclusion.

*Definition 7.* Given chromosome  $C = (\Sigma, X)$ , we say that  $\Sigma' \subseteq \Sigma$  is a  $\delta$ -chain of  $C$  if there exists some  $\delta$ -subsequence  $C'$  of  $C$  such that  $\Sigma' = \Sigma(C')$ . We say that  $C'$  is a *witness* to such a  $\delta$ -chain.

*Example 2.* When  $\delta = 2$ , the chromosome of Example 1 is comprised of two  $\delta$ -runs,  $\{(1, e)\}$  and  $\{(4, e), (5, d), (6, a), (8, b), (10, e), (11, d)\}$ . Examples of  $\delta$ -chains include  $\{d, e\}$  as witnessed by  $\{(4, e), (5, d)\}$ ,  $\{a, b, d, e\}$  as witnessed by  $\{(4, e), (5, d), (6, a), (8, b), (10, e), (11, d)\}$ , as well as  $\{d, a, b\}$  as witnessed by  $\{(5, d), (6, a), (8, b)\}$ .

*Definition 8.* Let  $\Sigma$  be the set of common COGs for two chromosomes  $C$  and  $D$ . A subset  $\Sigma' \subseteq \Sigma$  is a  $\delta$ -set of  $C$  and  $D$  if  $\Sigma'$  is a  $\delta$ -chain of both  $C$  and  $D$ . We say that a pair  $(C', D')$  *witnesses* the  $\delta$ -set if  $C'$  (resp.  $D'$ ) is a witness to the  $\delta$ -chain of  $C$  (resp.  $D$ ).

*Definition 9.* For two chromosomes  $C$  and  $D$ , a  $\delta$ -set  $\Sigma'$  is a  $\delta$ -team if it has a witness pair  $(C', D')$  such that no other  $\delta$ -set  $\Sigma^* \supset \Sigma'$  has a witness of the form  $(C^*, D^*)$  with  $C^* \supseteq C'$  and  $D^* \supseteq D'$ .

*Example 3.* Consider chromosomes  $C = \langle c**eda*b*ed \rangle$  and  $D = \langle ab*ba**c*dee*a \rangle$ . With  $\delta = 2$ , the non-trivial  $\delta$ -sets of  $C$  and  $D$  are  $\{a, b\}$ ,  $\{a, e\}$ ,  $\{d, e\}$  and  $\{a, d, e\}$ . Notice that  $\{a, d\}$  is not a  $\delta$ -set, even though  $\{a, d, e\}$  is. The non-trivial  $\delta$ -teams of  $C$  and  $D$  are:  $\{a, b\}$ ,  $\{d, e\}$  and  $\{a, d, e\}$ . Notice that  $\{d, e\}$  qualifies as a  $\delta$ -team due to witness  $C' = \{(e, 10), (d, 11)\}$  and  $D' = \{(d, 10), (e, 11), (e, 12)\}$ .

A  $\delta$ -team is then a maximal  $\delta$ -set with respect to subset inclusion of a witness. As it is a set of COGs, we informally call this a COG team. We note that the genes in the witnesses of a  $\delta$ -team may occur in different orders in the underlying chromosomes. Genomes have been known to undergo constant rearrangement and shuffling, hence the order of genes is not generally conserved.

## 3.2 Comparison to a Previous Model

Our model is a direct generalization of a previous model for gene teams which assumes a one-to-one orthologous relationship [1, 7]. Our definitions of  $\delta$ -chain,  $\delta$ -set and  $\delta$ -team reduce precisely to the original definitions, if applied to a pair of chromosomes with a one-to-one orthologous relationship. Unfortunately, many convenient properties in the restricted setting do not apply in general. Most significantly, the following properties are proven in the restricted setting [1]. If  $\Sigma_1$  and  $\Sigma_2$  are  $\delta$ -chains of  $C$  and  $\Sigma_1 \cap \Sigma_2 \neq \emptyset$ , then  $\Sigma_1 \cup \Sigma_2$  is also a  $\delta$ -chain. A similar statement follows for  $\delta$ -sets, and thus it is shown that the collection of  $\delta$ -teams forms a disjoint partition of the underlying genes. With the existence of paralogs, these properties may no longer hold. If we consider chromosome  $D$  from Example 3, we see that both  $\{a, b\}$  and  $\{a, e\}$  are  $\delta$ -chains with  $\delta = 2$ , yet  $\{a, b, e\}$  is not such a chain. We see that  $a$  is in  $\delta$ -team  $\{a, b\}$  as well as  $\{a, d, e\}$ . In general, these vastly different structural properties complicate the algorithmic search for  $\delta$ -teams.

## 4. ALGORITHMIC APPROACH

We consider two chromosomes  $C$  and  $D$  represented as sequences over a common alphabet  $\Sigma$  of COGs, as in Definition 1. We let  $m$  and  $n$  denote the number of genes in  $C$  and  $D$ , respectively, excluding genes which are not in the common alphabet. In this section, we present a divide-and-conquer algorithm which discovers each  $\delta$ -team for a given  $\delta$  in  $O(mn)$  time and  $O(m+n)$  space.

Our presentation is as follows. We begin in Section 4.1, with a comparison to the previous algorithm of [1], which applies only in the absence of paralogous genes. Section 4.2 contains the detailed presentation of our algorithm. A proof of correctness is given in Section 4.3, followed by an analysis of the time and space usage in Section 4.4. There are several possible forms of desired output which are discussed in Section 4.5 and an extension to circular chromosomes is given in Section 4.6.

### 4.1 Comparison to A Previous Algorithm

In the absence of paralogous genes, Bergeron, Corteel and Raffinot present an  $O((m+n) \log^2(m+n))$ -time algorithm to identify gene teams between two chromosomes [1]. Both that algorithm and ours rely on a divide-and-conquer approach which begins by breaking one of the chromosomes into subchromosomes separated by a gap greater than  $\delta$  between the position of relevant genes. The algorithm of Bergeron, Corteel and Raffinot, further relies on the fact that each gene in the first chromosome has at most one ortholog in a second chromosome. Thus, they can partition the second chromosome into two disjoint subsequences (though not necessarily subchromosomes) in step with the division of genes in the first chromosome. Thereafter, they recurse on independent subproblems.

With the existence of paralogous genes, the situation is quite different, as noted in Section 3.2. The COG teams do not necessarily form a partition of the COGs, so we cannot assume a linear bound on the cumulative cardinality of all such teams. Algorithmically, our approach differs from that of [1] at the division into subproblems. Though we will divide the first chromosome at gaps of greater than  $\delta$ , we are unable to divide the second chromosome in step with the first; genes from the second chromosome may have orthologs on several pieces of the first chromosome.

```

FINDTEAMS( $C, D$ )
  // Initialize Global Data
  GLOBALTIME  $\leftarrow 0$ 
  for each  $c$  in  $\Sigma$ 
    STAMP[ $c$ ]  $\leftarrow 0$ ; TEMP MARK[ $c$ ]  $\leftarrow 0$ 

  // Decompose problem based on initial runs of  $C$ 
  LOCALTIME  $\leftarrow$  MARKCOMMONALPHABET( $C, D$ )
   $C_{\text{rest}} \leftarrow C$ 
  repeat
     $C_{\text{first}} \leftarrow$  FINDFIRSTRUN( $C_{\text{rest}}, \text{LOCALTIME}$ )
     $C_{\text{rest}} \leftarrow C_{\text{rest}} \setminus C_{\text{first}}$ 
    FINDTEAMSRECURSE( $D, C_{\text{first}}$ )
  until ( $C_{\text{rest}} = \emptyset$ )

```

**Figure 1: The FINDTEAMS procedure.**

```

FINDTEAMSRECURSE( $A, B$ )
  LOCALTIME  $\leftarrow$  MARKCOMMONALPHABET( $A, B$ )
   $A_{\text{first}} \leftarrow$  FINDFIRSTRUN( $A, \text{LOCALTIME}$ )
   $A_{\text{rest}} \leftarrow A \setminus A_{\text{first}}$ 
  if  $A_{\text{rest}} = \emptyset$  then
    REPORTTEAM( $A, B$ ) // common alphabet
    // forms a  $\delta$ -team
  else
    repeat
      FINDTEAMSRECURSE( $B, A_{\text{first}}$ )
       $A_{\text{first}} \leftarrow$  FINDFIRSTRUN( $A_{\text{rest}}, \text{LOCALTIME}$ )
       $A_{\text{rest}} \leftarrow A_{\text{rest}} \setminus A_{\text{first}}$ 
    until ( $A_{\text{rest}} = \emptyset$ )

```

**Figure 2: The FINDTEAMSRECURSE procedure.**

## 4.2 Algorithm Presentation

Our approach is to split one of the original chromosome into its  $\delta$ -runs, based upon gaps of size greater than  $\delta$  between genes of common COGs. Then, a recursive subproblem can be defined for each such  $\delta$ -run, when paired with the *subsequence* of the entire other chromosome formed by taking all genes with orthologs in the  $\delta$ -run. A straightforward implementation of this recursion leads to an  $O(mn)$  time bound.

In order to achieve overall linear space usage, we must be careful in succinctly representing the subproblems, as the recursive depth is bounded only by  $O(m + n)$ . Therefore, we must use only constant space (on average) to represent any parameters and local variables on the recursive stack. We note the distinction made in Definition 4 between a *subsequence* and a *subchromosome*. Conceptually, each subproblem is defined based upon a pair of *subsequences* of the original chromosomes, since genes which are not common to the two subsequences would have been omitted at a higher level of the recursion. Representing each of those subsequences directly would require space proportional to the cardinality of such subsequences, at each recursive level. Instead, we represent the subproblem by the pair of *subchromosomes* which contain the *subsequences*. Each subchromosome can be represented in constant space, as a triple (*chrom, left, right*), where *chrom* is a reference to the original chromosome and *left* and *right* are references to the leftmost and rightmost genes of  $A$ , respectively.

Unfortunately, another subtlety arises due to the need to

```

// Relies on use of global data:
// GLOBALTIME, TEMP MARK[ ], STAMP[ ]
MARKCOMMONALPHABET( $A, B$ )
  GLOBALTIME = GLOBALTIME + 1
  for each  $g$  in  $A$ 
    Let  $c$  be the COG of  $g$ 
    TEMP MARK[ $c$ ]  $\leftarrow$  GLOBALTIME
  for each  $g$  in  $B$ 
    Let  $c$  be the COG of  $g$ 
    if TEMP MARK[ $c$ ] = GLOBALTIME then
      STAMP[ $c$ ]  $\leftarrow$  GLOBALTIME
  return GLOBALTIME

```

**Figure 3: The MARKCOMMONALPHABET procedure.**

```

FINDFIRSTRUN( $A, \text{TIMESTAMP}$ )
   $endRun \leftarrow$  first gene in  $A$  with STAMP[ $c$ ]  $\geq$  TIMESTAMP
   $nextGene \leftarrow$  gene which follows  $endRun$  in  $A$ 
  while ( $nextGene$  is well defined
    AND  $\Delta(endRun, nextGene) \leq \delta$ ) do
    // check whether  $nextGene$  is in common alphabet,
    // and thus whether to extend the run
    if STAMP[ $nextGene$ ]  $\geq$  TIMESTAMP then
       $endRun \leftarrow nextGene$ 
    // in either case continue advancing
    increment  $nextGene$  with respect to  $A$ 
  return subchromosome up to and including  $endRun$ 

```

**Figure 4: The FINDFIRSTRUN procedure.**

discern the underlying subsequences within the represented subchromosomes. This seems to require a representation of the locally common alphabet at each recursive step, whether recomputed or sent as a parameter. Again, we must be careful to achieve overall linear space usage. Our solution is to implicitly represent all locally common alphabets of the active recursion simultaneously, through the use of the following *global* structures. There exists an integer GLOBALTIME, and two integer arrays STAMP[ ] and TEMP MARK[ ], with entries for each member of  $\Sigma$ . Since the local alphabet at one level of recursion is a subset of the local alphabet at the parent level, the timestamps for common COGs can be incremented to denote the local alphabet, while not violating the parent's ability to discern its own alphabet.

Finally, we note one additional subtlety in our approach. It is important in the analysis that we break a subchromosome into *all* appropriate  $\delta$ -runs at a given level of our recursion. In [1], only the first run of a chromosome is broken off at a given recursive level; the remainder is left within a single subproblem, though likely broken at deeper levels of recursion. Our time analysis could not be applied if we tried a similar approach here, due to our higher overhead cost at each level in recomputing the effective local alphabet. By instead insisting that a subproblem be broken into all of its  $\delta$ -runs, we are assured that the following recursive level can only result in further subproblems by a split of the opposite chromosome. This leads us to an alternating, double recurrence, given in Section 4.4.

A detailed algorithm description is given in Figures 1–4, and the analysis of the algorithm follows in Section 4.3–4.4.

### 4.3 Algorithm Correctness

We begin with several lemmas involving the use of the global arrays in representing common alphabets during the recursion.

LEMMA 1. *At the onset of MARKCOMMONALPHABET, for each COG  $c$ ,  $\text{STAMP}[c] \leq \text{TEMPMARK}[c] \leq \text{GLOBALTIME}$ .*

PROOF.  $\text{STAMP}[c]$ ,  $\text{TEMPMARK}[c]$ , and  $\text{GLOBALTIME}$  are initialized to 0 for each  $c$  in  $\text{FINDTEAMS}$ . After that, they are only modified within  $\text{MARKCOMMONALPHABET}$ . The invariant is evident from Figure 3.  $\square$

LEMMA 2. *A call to MARKCOMMONALPHABET( $A, B$ ) effects array  $\text{STAMP}[\ ]$  as follows:*

$$\text{STAMP}[c] = \begin{cases} \text{GLOBALTIME} & \text{if } c \in \Sigma(A) \cap \Sigma(B) \\ \text{left unchanged} & \text{otherwise} \end{cases}$$

PROOF. Evident from Figure 3 and Lemma 1.  $\square$

LEMMA 3. *A call to FINDTEAMSRECURSE( $A, B$ ) effects  $\text{STAMP}[\ ]$  as follows. The value  $\text{STAMP}[c]$  remains unchanged for all  $c \notin \Sigma(A) \cap \Sigma(B)$ , and may only be increased for  $c \in \Sigma(A) \cap \Sigma(B)$ .*

PROOF. The array  $\text{STAMP}[\ ]$  is only modified within calls to  $\text{MARKCOMMONALPHABET}$ . The call to  $\text{MARKCOMMONALPHABET}$  made from the first line of the  $\text{FINDTEAMSRECURSE}$  routine respects our claim. This follows directly from Lemmas 1 and 2 with the observation that  $\text{GLOBALTIME}$  increases.

If the call to  $\text{FINDTEAMSRECURSE}$  invokes recursion from within the “repeat” loop, we can prove this lemma by induction on the maximum recursive depth. By inspection of Figure 4, we can see that each assignment to  $A_{\text{first}}$  is a subchromosome of  $A$ . Thus,  $\Sigma(A_{\text{first}}) \cap \Sigma(B) \subseteq \Sigma(A) \cap \Sigma(B)$ . By induction, this means that  $\text{STAMP}[c]$  remains unchanged within the recursion for  $c \notin \Sigma(A) \cap \Sigma(B) \supseteq \Sigma(A_{\text{first}}) \cap \Sigma(B)$ .  $\square$

LEMMA 4. *Whether invoked from FINDTEAMS( $X, Y$ ) or FINDTEAMSRECURSE( $X, Y$ ), it must be that upon onset of  $\text{FINDFIRSTRUN}$ ,*

$$\begin{aligned} \text{STAMP}[c] &\geq \text{TIMESTAMP} && \text{if } c \in \Sigma(X) \cap \Sigma(Y) \\ \text{STAMP}[c] &< \text{TIMESTAMP} && \text{if } c \notin \Sigma(X) \cap \Sigma(Y) \end{aligned}$$

PROOF. After the first call to  $\text{MARKCOMMONALPHABET}$ , the condition is true due to Lemmas 1 and 2 with the observation that  $\text{GLOBALTIME}$  increases. It remains true throughout the “repeat” loop due to Lemma 3.  $\square$

LEMMA 5. *For a call to  $\text{FINDFIRSTRUN}(A, \text{TIMESTAMP})$ , let  $A'$  be the returned subchromosome. Define  $A^*$  to be the subsequence of  $A$  containing those COGs with  $\text{STAMP}[c] \geq \text{TIMESTAMP}$ . If  $A^* \neq \emptyset$  then  $A' \cap A^*$  is the first  $\delta$ -run of  $A^*$ .*

PROOF. We refer to the description of  $\text{FINDFIRSTRUN}$  from Figure 4. If  $A^* = \emptyset$ , the claim is trivial. Otherwise, we designate the first  $\delta$ -run of  $A^*$  as genes  $g_1, g_2, \dots, g_k$  for some  $k$ , noting that  $\Delta(g_i, g_{i+1}) \leq \delta$  for all  $1 \leq i < k$ .

(*first  $\delta$ -run of  $A^*$* )  $\subseteq (A' \cap A^*)$ : We show by induction that for each  $1 \leq i \leq k$ ,  $\text{endRun}$  is at some point equal to  $g_i$ .  $\text{endRun}$  is initially set to  $g_1$ . If we assume  $\text{endRun} = g_i$

at some point for each  $1 \leq i < k$ , we show that it is so for  $g_{i+1}$ . Since  $\Delta(g_i, g_{i+1}) \leq \delta$ , the while loop will advance until  $\text{nextGene}$  is equal to  $g_{i+1}$ . Then, since  $\text{STAMP}[g_{i+1}] \geq \text{TIMESTAMP}$ ,  $\text{endRun}$  will be set to  $g_{i+1}$ .

( *$A' \cap A^*$* )  $\subseteq$  (*first  $\delta$ -run of  $A^*$* ): By the previous paragraph, there is a point at which  $\text{endRun}$  is equal to  $g_k$ . We claim that  $\text{endRun}$  cannot be updated after that point. The only way in which  $\text{endRun}$  is updated would be if there exists some  $g$  with  $\Delta(g_k, g) \leq \delta$  and with  $\text{STAMP}[g] \geq \text{TIMESTAMP}$ . Yet if such a  $g$  existed, then  $g_k$  could not end a  $\delta$ -run of  $A^*$ .  $\square$

LEMMA 6. *For each call to  $\text{FINDTEAMSRECURSE}(A, B)$ ,  $\Sigma(A) \cap \Sigma(B)$  is a  $\delta$ -chain of  $B$ .*

PROOF. For a call to  $\text{FINDTEAMSRECURSE}(A, B)$ , we consider the routine from which it was invoked. From that caller there must exist subchromosomes which we will denote as  $A_{\text{caller}}$  and  $B_{\text{caller}}$  such that the following are true.  $A = B_{\text{caller}}$ ;  $B$  is the first  $\delta$ -run of  $A_{\text{caller}}$  when restricted to the common alphabet  $\Sigma(A_{\text{caller}}) \cap \Sigma(B_{\text{caller}})$ . This second property follows from Lemma 5. This means that  $\Sigma(A_{\text{caller}}) \cap \Sigma(B_{\text{caller}}) = \Sigma(A_{\text{caller}}) \cap \Sigma(A)$  is a  $\delta$ -chain of  $B$ . Furthermore, since  $B$  is a subset of  $A_{\text{caller}}$ , the witness to the above  $\delta$ -chain cannot contain any COGs from  $\Sigma(A_{\text{caller}}) \setminus \Sigma(B)$ , and thus it must be that  $\Sigma(B) \cap \Sigma(A)$  is a  $\delta$ -chain of  $B$ .  $\square$

LEMMA 7. *Given (sub)chromosomes  $A$  and  $B$ , if  $A$  consists of  $\delta$ -runs  $A_1, A_2, \dots, A_k$ , then*

$$\begin{aligned} \{\Sigma' : \Sigma' \text{ is a } \delta\text{-team of } A \text{ and } B\} = \\ \bigcup_{1 \leq i \leq k} \{\Sigma' : \Sigma' \text{ is a } \delta\text{-team of } A_i \text{ and } B\} \end{aligned}$$

PROOF. *Leftside  $\subseteq$  Rightside*: Assume  $\Sigma'$  is a  $\delta$ -team of  $A$  and  $B$ . From Definition 9, we consider any witness pair  $A'$  and  $B'$  which are  $\delta$ -subsequences of  $A$  and  $B$  respectively, with  $\Sigma' = \Sigma(A') = \Sigma(B')$ . By Definition, if  $A'$  is not itself a  $\delta$ -run of  $A$ , it must be a subset of some  $\delta$ -run  $A_i$  of  $A$ . Thus  $\Sigma'$  is a  $\delta$ -set of  $A_i$  and  $B$ , as witnessed by pair  $(A', B')$ . It must be that  $\Sigma'$  is a  $\delta$ -team of  $A_i$  and  $B$ . Assume for contradiction there exists some  $\Sigma^* \supset \Sigma'$  with witness pair  $A^*$  and  $B^*$  such that  $A' \subseteq A^* \subseteq A_i$  and  $B' \subseteq B^*$ . Such a  $\Sigma^*$  must be a  $\delta$ -set of  $A$  and  $B$ , yet this contradicts the original assumption that  $\Sigma'$  is a  $\delta$ -team of  $A$  and  $B$ .

*Rightside  $\subseteq$  Leftside*: Assume  $\Sigma'$  is a  $\delta$ -team of  $A_i$  and  $B$ , and witnessed by  $A'$  and  $B'$ . This same witness pair demonstrates that  $\Sigma'$  is a  $\delta$ -set of  $A$  and  $B$ . It must be that  $\Sigma'$  is a  $\delta$ -team of  $A$  and  $B$ . Assume for contradiction that there exists some  $\Sigma^* \supset \Sigma'$  with witness pair  $A^*$  and  $B^*$  such that  $A' \subseteq A^* \subseteq A$  and  $B' \subseteq B^*$ . Since there is a gap of greater than  $\delta$  between genes from separate  $\delta$ -runs, it must be that such  $A^* \subseteq A_i$  and thus  $\Sigma^*$  a  $\delta$ -set of  $A_i$  and  $B$ . This contradicts the original assumption that  $\Sigma'$  is a  $\delta$ -team of  $A_i$  and  $B$ .  $\square$

LEMMA 8. *For the recursion which results from a given call to  $\text{FINDTEAMSRECURSE}(A', B')$ :*

- (1) *If  $\text{REPORTTEAM}(A, B)$  is called,  $\Sigma(A) \cap \Sigma(B)$  forms a  $\delta$ -team of  $A'$  and  $B'$ .*
- (2) *If  $\Sigma'$  is a  $\delta$ -team for  $A'$  and  $B'$ ,  $\text{REPORTTEAM}(A, B)$  will be called for some  $A$  and  $B$  such that  $\Sigma' = \Sigma(A) \cap \Sigma(B)$ .*

PROOF. We prove this claim by induction on the depth of recursion. If the call does not invoke any recursion, then it is because the initial call to `FINDFIRSTRUN` determined, as shown in Lemma 5, that  $A'$  is  $\delta$ -run relative to the alphabet  $\Sigma(A') \cap \Sigma(B')$ . Since  $B'$  is known to be a  $\delta$ -chain as well, by Lemma 6, then  $\Sigma(A') \cap \Sigma(B')$  is a  $\delta$ -set of  $A'$  and  $B'$ . Since the witnesses include all of  $A'$  and  $B'$ , this must in fact be a  $\delta$ -team. Furthermore, there could not possibly be any other  $\delta$ -team for  $A'$  and  $B'$  as its witnesses would necessarily be a subset of  $A'$  and  $B'$  which we have just seen as witnesses to a  $\delta$ -team.

If recursion is invoked, `FINDTEAMSRECURSE`( $B', A'_i$ ) is called for each  $\delta$ -run of  $A'$ . Applying induction together with Lemma 7 completes the claim.  $\square$

THEOREM 1. For a given call to `FINDTEAMS`( $C, D$ ):

- (1) If `REPORTTEAM`( $A, B$ ) is called,  $\Sigma(A) \cap \Sigma(B)$  forms a  $\delta$ -team of  $C$  and  $D$ .
- (2) If  $\Sigma'$  is a  $\delta$ -team for  $C$  and  $D$ , `REPORTTEAM`( $A, B$ ) will be called for some  $A$  and  $B$  such that  $\Sigma' = \Sigma(A) \cap \Sigma(B)$ .

PROOF. The call to `FINDTEAMS`( $C, D$ ) results in a call to `FINDTEAMSRECURSE`( $D, C_i$ ) for each  $\delta$ -run of  $C$ . The claim is a direct result of Lemmas 7 and 8.  $\square$

#### 4.4 Analysis of Time and Space Usage

In this section, we let  $A$  and  $B$  denote subchromosomes of the original chromosomes  $C$  and  $D$  (though not necessarily respectively), and  $a$  and  $b$  the respective number of genes in  $A$  and  $B$ .

We begin by examining the running time of a call to `FINDTEAMSRECURSE`( $A, B$ ). We let  $\Sigma'$  denote  $\Sigma(A) \cap \Sigma(B)$ . We let  $k$  denote the number of  $\delta$ -runs of  $A$  over  $\Sigma'$ , and  $a_1, \dots, a_k$  the cardinalities of those  $\delta$ -runs. It is easily seen that the running time for this procedure satisfies the following recurrence, for some constant  $c > 0$ .

$$T(a, b) \leq \begin{cases} c(a+b) & \text{when } k = 1 \\ c(a+b) + \sum_{1 \leq i \leq k} T(b, a_i) & \text{when } k \geq 2 \end{cases}$$

The overhead cost of  $c(a+b)$  is due to the call made to `MARKCOMMONALPHABET`( $A, B$ ) and the cumulative time spent on the  $k$  calls to `FINDFIRSTRUN`. When  $k \geq 2$ , the summation is due to the recursion.

LEMMA 9.  $T(a, b) = O(ab)$

PROOF. We prove a more specific claim, that there exists a constant  $c' > 0$  such that  $T(a, b) \leq c'ab - 2ca - 3cb$ . We prove this by induction on  $(a+b)$ .

As a base case, when  $a = b = 1$ ,  $k$  must be 1 and thus the running time is bounded by a constant. By choosing  $c'$  to be sufficiently large, the claim can be made true. In general, we consider larger values of  $(a+b)$ . For a given invocation of the procedure, we consider two possible cases, depending on whether  $k = 1$ . When  $k = 1$ , then the recurrence dictates that  $T(a, b) \leq c(a+b)$ . By choosing  $c' \geq 8c$ , we see that

$$\begin{aligned} T(a, b) &\leq c(a+b) = (3ca - 2ca) + (4cb - 3cb) \\ &\leq 4c(a+b) - 2ca - 3cb \leq 4c(2ab) - 2ca - 3cb \\ &\leq c'ab - 2ca - 3cb \end{aligned}$$

When  $k > 1$ , we apply induction, to see that

$$\begin{aligned} T(a, b) &\leq c(a+b) + \sum_{1 \leq i \leq k} T(b, a_i) \\ &\leq c(a+b) + \sum_{1 \leq i \leq k} (c'ba_i - 2cb - 3ca_i) \\ &\leq c(a+b) - 2kcb + (c'b - 3c)a \\ &= c'ba - 2ca - (2k-1)cb \\ &\leq c'ba - 2ca - 3cb \end{aligned}$$

The third inequality holds since  $(c'b - 3c) > 0$  and  $\sum_i a_i \leq a$ . The final inequality holds as  $k \geq 2$ .  $\square$

THEOREM 2. `FINDTEAMS`( $C, D$ ) runs in  $O(mn)$  time.

PROOF. The call to `FINDTEAMS`( $C, D$ ) results in a call to `FINDTEAMSRECURSE`( $D, C_i$ ) for each of  $k$   $\delta$ -runs of  $C$ , with  $m_i$  the size of  $C_i$ . Therefore, by Lemma 9, the overall time is  $O(m_1n + m_2n + \dots + m_kn) = O(mn)$ .  $\square$

THEOREM 3. `FINDTEAMS`( $C, D$ ) uses  $O(m+n)$  space.

PROOF. Global arrays `STAMP`[] and `TEMPMARK`[] use space proportional to  $\Sigma = O(m+n)$ . The only additional space we must account for is that required for the recursion. The parameters and local variables within a single level of the recursion requires constant space. Since at least one gene is removed at each recursive level, then can be at most  $O(m+n)$  active calls at any one time.  $\square$

#### 4.5 Variants on Algorithm Output

Theorem 1 guarantees that for each  $\delta$ -team, the procedure `REPORTTEAM`( $A, B$ ) will be called, where  $A$  and  $B$  are subchromosomes whose common alphabet defines the  $\delta$ -team. As of yet, we have not explicitly declared such a routine. Our running time analysis of Section 4.4 holds true so long as the time spent within a call to `REPORTTEAM`( $A, B$ ) is  $O(a+b)$ . By using a technique similar to `MARKCOMMONALPHABET`, the  $\delta$ -team as well as its witness, can be output within  $O(a+b)$  time.

In Section 4.1, we noted that COG teams do not necessarily form a partition, and thus that the cumulative cardinality may be super-linear. If the output is to be considered as part of the memory usage, then an  $O(m+n)$  space bound no longer applies. In fact, the number of COG teams is not necessarily bounded by  $O(m+n)$ .

One remedy is to flush the output to disk as it is generated, in which case it need not be considered as part of the memory usage. If a strict linear space bound is desired, an alternative is to output information for the most “meaningful”  $\delta$ -teams. For example, for each COG  $c$ , we could track the maximal cardinality  $\delta$ -team which includes  $c$ . To keep the linear bound on the space usage, a candidate  $\delta$ -team can be stored via the two triples which define the containing subchromosomes; the resulting  $\delta$ -teams could then be reconstructed at the conclusion.

#### 4.6 Circular Chromosomes

We note briefly that our algorithm easily extends to the case of circular chromosomes with appropriate modification to our representation of a subchromosome and the procedure `FINDFIRSTRUN`. If a gap of greater than  $\delta$  is found, a circular chromosome can be viewed linearly.

Genome	Number of genes		
	overall	from an identified COG	from an identified COG common to both genomes
<i>E. coli</i>	4290	3289 (76.7%)	2332 (54.4%)
<i>B. subtilis</i>	4101	2818 (68.7%)	2339 (57.0%)

**Table 1: Percentage of overall genes identified in clusters of orthologous groups**

COG type	No. COGs	Genes assigned to such COGs	
		<i>E. coli</i>	<i>B. subtilis</i>
one-to-one	578	578	578
one-to-many	299	517	568
many-to-many	260	1237	1193

**Table 2: Distribution of orthologous groups in the compared genomes**

## 5. EXPERIMENTAL RESULTS

In this section, we present preliminary results obtained using our methodology. Our immediate goal was to demonstrate the feasibility of our algorithmic approach as well as to explore the strengths and limitations of the methodology. We perform comparative analysis of two prokaryote genomes, *E. coli* K12 and *B. subtilis*.

The analysis process is comprised of two distinct steps. First, the orthologous relationships among genes must be analyzed, so that clusters of orthologous groups (COGs) can be identified. Second, the algorithm described in Section 4 will be applied to identify all  $\delta$ -teams for a chosen  $\delta$ , which represent the conserved gene clusters in a biological context. The first task of identifying orthologous groups is known to be a difficult problem. The difficulty lies in the fact that the sequence similarities of genes do not correspond exactly with their evolutionary relationship. For instance, the paralogs of a gene that predate the species split, are easily confused with the true orthologs. Many approaches have been proposed for this problem, from the simple bidirectional best hit method, to more advanced programs, such as InParanoid [10]. For the ease of experimentation, we took advantage of existing data from NCBI’s database [8]. The proteins in the downloaded files are already annotated with the COG number, which is uniquely assigned to each of 3307 COGs currently identified from the analysis of 43 complete genomes at NCBI. All proteins with the same COG number will be considered as orthologs if in different chromosomes, or paralogs if in the same chromosome. This method of classifying orthologs is different from the one used in previous papers [7, 9], namely the bidirectional best hit method which guarantees one-to-one orthologous relationships. The advantages of our method will be shown below.

A significant percentage of genes from *E. coli* K12 and *B. subtilis* are from COGs common to these two genomes. Statistics of our COG assignment are given in Table 1. It can be seen that more than half of the genes in two genomes belongs to the shared COGs. As we have explained in Section 2, one gene, say  $g$ , in the first genome can have one or multiple orthologs in another genome if the ortholog of  $g$  formed paralogs in the second genome. The consequent types of COG will be named as one-to-one or one-to-many, depending on the number of orthologs of  $g$  in the second genome. If  $g$  itself duplicated and resulted in new paralogs during evolution, the COG associated with  $g$  will include

both orthologs and paralogs of  $g$  in two genomes, and it will be thus called a many-to-many COG. The distribution of COGs based on orthologous relationships is shown in Table 2. Of all the COGs involved in both genomes, approximately half exhibit a one-to-many or many-to-many relationship. In terms of the underlying common genes, more than two thirds belong to these one-to-many or many-to-many COGs. This further substantiates the importance of considering more general orthologous relationships when identifying conserved gene clusters. The original gene team approach was limited in application to one-to-one COGs, identified by a bidirectional best hit method.

Based on the given COG assignments between *E. coli* and *B. subtilis*, our algorithm identified 85  $\delta$ -teams for  $\delta = 1000$  base pairs. Of these teams, 65 have two members; 16 have three members; 3 have four members and one has 21 members. More complete data for this experiment is available at <http://euler.slu.edu/~goldwasser/cogteams/data>. It is interesting to note that the current COG database contains many COGs whose function remain undetermined; perhaps by examining the genes in the same teams of these uncharacterized COGs, we may be able to find some clues of the function of these COGs. The COG teams that contain the uncharacterized COGs are shown here. In addition, all the  $\delta$ -teams but one whose sizes are larger than four are also shown at Table 3. The largest COG team contains 21 ribosomal proteins, which is similar to the findings of [7].

It should be noted that the value of parameter  $\delta$  affects the output. The larger the  $\delta$  value is, the more COG teams will be identified. To reduce false positives, we chose a relatively small value of 1000 base pairs in our experiment. We believe it may be necessary to adjust this parameter by manual inspection of the results in general. However, the most significant COG teams seem not dependent on the exact value of  $\delta$ .

## 6. DISCUSSION

We compare our experimental approach for identifying conserved gene clusters to previous approaches, especially that of [1, 7] which we have generalized. Most significant is that our method can consider multiple paralogs of genes in a genome. We believe this extension of model is already essential for the examination of many prokaryote genomes, and even more so for the analysis of eukaryote genomes, which are generally thought to contain many paralogs. In practice,

Size	Members
2	Predicted S-adenosylmethionine-dependent methyltransferase(COG0275), Uncharacterized protein conserved in bacteria (COG2001)
2	Uncharacterized protein conserved in bacteria(COG0718), Recombinational DNA repair protein (RecF pathway)(COG0353)
2	Uncharacterized enzyme of thiazole biosynthesis(COG2022), Dinucleotide-utilizing enzymes involved in molybdopterin and thiamine biosynthesis(COG0476)
4	Phosphoribosyl-ATP pyrophosphohydrolase(COG0140), Imidazoleglycerol-phosphate synthase(COG0107), Glutamine amidotransferase(COG0118), Phosphoribosylformimino-5-aminoimidazole carboxamide ribonucleotide (ProFAR) isomerase(COG0106)
4	Ribosomal protein L11(COG0080), Preprotein translocase subunit SecE(COG0690), Ribosomal protein L1(COG0081), Transcription antiterminator(COG0250)
4	F0F1-type ATP synthase, subunit c/Archaeal/vacuolar-type H <sup>+</sup> -ATPase, subunit K(COG0636), F0F1-type ATP synthase, subunit b(COG0711), F0F1-type ATP synthase, subunit a(COG0356), F0F1-type ATP synthase, delta subunit (COG0712)
21	Ribosomal proteins: L24(COG0198), L22(COG0091), L29(COG0255), S10(COG0051), S5(COG0098), L18(COG0256), S19(COG0185), L23(COG0089), L5(COG0094), L3(COG0087), L15(COG0200), S3(COG0092), L6P/L9E(COG0097), L4(COG0088), L30/L7E(COG1841), L16/L10E(COG0197), S8(COG0096), L2(COG0090), S14(COG0199), L14(COG0093), S17(COG0186)

**Table 3:**  $\delta$ -teams including at least one uncharacterized COG or four characterized COGs, with  $\delta = 1000$  bps

this extended capability of algorithm eliminates the need of building orthologous groups from scratch as we can import data directly from NCBI COG database. Methods that depend on one-to-one orthologous relationship have no such advantage.

We do note that our current reliance of existing COG databases is a limitation. For example, we had thought that the method would be able to reconstruct some known operons, such as the *trp* operon. However, it turns out that most of the *trp* operon genes are not represented in the COG database. Similar phenomena are observed for gene clusters in purine biosynthesis pathway, glycolysis pathway, etc. Our methodology does not explicitly require this reliance on existing COG databases, rather those databases were a valuable resource to us in rapidly developing preliminary experiments. Given more time, common COGs for given genomes could be identified by other methods for further experimentation.

One drawback to our generalized model is that it does not extend efficiently to the analysis of more than simply pairwise genome comparison. Without paralogs, the divide-and-conquer algorithm extends with relative efficiency. With our generalization, our algorithm would have a worst case time complexity of  $O(m_1 m_2 \dots m_n)$ , where  $m_i$  is the number of genes of the  $i$ -th chromosome from common COGs.

Finally, a limitation of both our work and the previous gene team model is in the interpretation of the results. We have no automated way to assess the true biological significance of identified  $\delta$ -teams. Rather we expect this tool to be used by biologists to quickly identify potential conserved gene clusters, deserving of further study by other means. However, it might be possible to assign the significance scores for  $\delta$ -teams in light of the recent progress in constructing the statistical test for gene clustering [4].

## 7. CONCLUSION

The identification of conserved gene clusters is important to many biological problems, such as the genome comparative mapping, the study of transcriptional neighborhood, the prediction of gene functions, etc. We developed a general mathematical model and algorithm for the problem of pairwise conserved gene cluster identification. We obtained the initial results from the analysis of two bacterial genomes: *E. coli* and *B. subtilis*.

We realize that the study of gene clusters is an exciting and in fact, open-ended area. We would like to point out several avenues of furthering the current project. The most obvious and interesting one is to apply these techniques to find biologically useful results, for instance, the identification of uncharacterized COGs, Secondly, it is important to distinguish between the true gene clusters due to evolutionary pressures and those simply due to the common ancestry. No information is provided by the present algorithm, therefore it seems necessary to develop or import statistical tests to assess the significance of  $\delta$ -teams identified, and in particular, the effect of  $\delta$  on the results. The last long-term goal would be to generalize our algorithm or develop a new one, if necessary, for comparison on more than two chromosomes; or alternatively to show that the average-case running time is in fact adequately fast in practice.

## 8. ACKNOWLEDGMENTS

Much of this research took place while the authors were at Loyola University Chicago. Michael Goldwasser's research is supported, in part, by the National Science Foundation through grant CCR-0208987. The authors are grateful to Howard Laten and Bryan Pickett for helpful conversations, and to Mathieu Raffinot for helpful comments.

## 9. REFERENCES

- [1] A. Bergeron, S. Corteel, and M. Raffinot. The algorithmic of gene teams. In *Proc. Second Annual Workshop on Algorithms in Bioinformatics*, volume 2452 of *Lecture Notes in Computer Science*, pages 464–476, Rome, Italy, Sept. 2002. Springer-Verlag.
- [2] T. Dandekar, B. Snel, M. Huynen, and P. Bork. Conservation of gene order: a fingerprint of proteins that physically interact. *Trends in Biochemical Sciences*, 23(9):324–328, 1998.
- [3] G. Didier. Common intervals of two sequences. In *Proc. Annual Workshop on Algorithms in Bioinformatics 2003*, volume 2812 of *Lecture Notes in Computer Science*, pages 17–24, 2003.
- [4] D. Durand and D. Sankoff. Tests for gene clustering. *Journal of Computational Biology*, 10:453–482, 2003.
- [5] W. Fujibuchi, H. Ogata, H. Matsuda, and M. Kanehisa. Automatic detection of conserved gene clusters in multiple genomes by graph comparison and P-quasi grouping. *Nucleic Acids Research*, 28(20):4029–4036, 2000.
- [6] S. Heber and J. Stoye. Algorithms for finding gene clusters. In *Proc. Annual Workshop on Algorithms in Bioinformatics 2001*, volume 2149 of *Lecture Notes in Computer Science*, pages 252–263, 2001.
- [7] N. Luc, J.-L. Risler, A. Bergeron, and M. Raffinot. Gene Teams: a new formalization of gene clusters for comparative genomics. *Computational Biology and Chemistry*, 27:59–67, 2003.
- [8] National Center for Biotechnology Information. Phylogenetic classification of proteins encoded in complete genomes, 2003. <http://www.ncbi.nih.gov/COG>.
- [9] R. Overbeek, M. Fonstein, M. D’Souza, G. D. Pusch, and N. Maltsev. The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences USA*, 96(6):2896–2901, 1999.
- [10] M. Remm, C. E. V. Storm, and E. L. L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *Journal of Molecular Biology*, 314:1041–1052, 2001.
- [11] P. T. Spellman and G. M. Rubin. Evidence for large domains of similarly expressed genes in the drosophila genome. *Journal of Biology*, 1:5, 2002.
- [12] R. L. Tatusov, E. V. Koonin, and D. J. Lipman. A genomic perspective on protein families. *Science*, 278:631–637, 1997.
- [13] R. L. Tatusov, D. A. Natale, I. V. Garkavtsev, T. A. Tatusova, U. T. Shankavaram, B. S. Rao, B. Kiryutin, M. Y. Galperin, N. D. Fedorova, and E. V. Koonin. The COG database: new developments in phylogenetic classification of proteins from complete genomes. *Nucleic Acids Research*, 29(1):22–28, 2001.
- [14] T. Uno and M. Tagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.